

Visual COBOL チュートリアル

COBOL 開発:ステップバイステップチュートリアル

Visual Studio 2022

1 目的

Visual COBOL for Visual Studio 2022 は、マイクロソフト社の最新開発環境である Visual Studio 2022 の IDE(統合開発環境)上で COBOL のアプリケーション開発を行うための製品です。 COBOL プログラマーが既存の COBOL 資産を Windows 環境で活用するだけでなく、COBOL 言語のプログラミング経験のないプログラマーが初めて COBOL アプリケーション開発を行う場合に最適な製品です。

このドキュメントは、Visual COBOL for Visual Studio 2022 を学ぶためのステップバイステップのチュートリアルです。

2 前提

- 本チュートリアルで使用したマシン OS : Windows 11
- Visual COBOL 10.0 for Visual Studio がすでにインストールされていること
 インストール手順は以下の FAQ サイトを参照してください。
- https://support.amc.rocketsoftware.co.jp/SupportInf/amc_faqpublic.aspx?VC01002
- プログラミングの基礎知識を有していること
- Windows の基本操作を理解していること

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダに解凍しておいてください。 サンプルプログラムのダウンロード



内容

- 1 目的
- 2 前提
- 3 チュートリアル
 - 3.1 Visual Studio の IDE に慣れてみよう
 - 3.2 はじめての Visual COBOL プロジェクト
 - 3.3 ファイルの入出力



3 チュートリアル

3.1 Visual Studio の IDE に慣れてみよう

- 1) Windows のスタートメニューから Visual Studio 2022 をクリックします。
- 2) Visual Studio を起動して任意の COBOL プロジェクトを作成すると以下のような画面が表示されます。



Microsoft Visual Studio 2022 の IDE は、メニューバー、ツールバー、左、下または右にドッキングまたは自動的に非 表示になる各種ツールウィンドウ、エディター領域など、複数の要素で構成されます。IDE 内の要素の配置は、適用した設 定とその後に加えたカスタマイズ内容によって異なります。

Visual Studio 2022 のソリューションとプロジェクトには、アプリケーションの作成に必要な参照、データ接続、フォルダ、お よびファイルを表す項目が含まれています。 ソリューションには複数のプロジェクトを含めることができ、プロジェクトには、通常、 複数の項目が含まれます。 ソリューションエクスプローラーには、ソリューション、それらのプロジェクト、そのプロジェクト内の項 目が表示されます。 ソリューションエクスプローラーを使用すると、編集するファイルを開く、プロジェクトに新規ファイルを追加 する、ソリューション、プロジェクト、および項目のプロパティを表示するなどの操作を実行できます。



Visual Studio 2022 のソースコードエディターには、COBOL 予約語とデータ名や手続き名などの利用者語を色分け表示や、COBOL スニペットなど COBOL 言語固有の機能拡張が含まれます。ソースコードを入力するとバックグラウンドチ

ェックを実行して、赤の波線でエラー箇所を強調表示します。 そのエラー箇所にマウスポインタを移動することでエラー内容の確認、定義への移動、他の参照検索などの操作が可能です。

Program1.cbl 🕂			~ \$
CBL ConsoleHello	✓ ^A ^C	Working-Storage Section 🔹	÷
	identification division. program-id. Program1.		
ŀ	environment division. configuration section.		
-	data division. working-storage section.		2
Ť	procedure division. soback.		
	end program Program1.		
			.
100 % 🔹 🥥	問題は見つかりませんでした) 行: 1 文字: 1	SPC CRLF

Visual Studio 2022 のビルド構成では、プラットフォームの選択、プロジェクトまたはソリューションのビルド方法を定義しま す。プロジェクトタイプごとに、デバッグとリリースのデフォルト構成があり、独自の構成を作成することも可能です。コンソールウ ィンドウにはビルド時のメッセージやアプリケーションのコンソール出力等が表示されます。問題ウィンドウには、不正な構文、 キーワードのスペルミス、型の不一致などのコンパイルエラーが表示されます。



ビルドしたアプリケーションは、実行時の論理エラーやセマンティックエラーなどの問題を検出して修正するために、デバッガーを 使用します。 Visual Studio 2022 のデバッガーは、コードのステップ実行、様々な条件を設定したブレークポイントで実 行、変数ウィンドウやウォッチ式などのツールを使用してローカル変数やその他の関連データを調べることができます。

0	data division. working-storage section. 01 str pic x(10). procedure division. move "おいうえお" to st display str.]	r.			
	soback. end program Program1.				
100 % 🔹 🕻	≥ 問題は見つかりませんでした <		þ	コピー(Y)	Ctrl+C
自動		•		値の編集(E)	
検索 (Ctrl+E)	$\rho \to [\bullet]$	検索の詳細度:		値のコピー(V)	
名前	値	種類	62	ウォッチの追加(W)	
🤣 str	あいうえお	Q 表示 ▼ PIC X(10)	62	並列ウォッチの追加(R)	
			4	すべて選択(A)	Ctrl+A
			٠	トレースポイントの挿入(T)	
				16 進数で表示(H)	



デバッグが完了したアプリケーションは、Windows インストーラーを使用するか、ファイルを手動でコピーして、本番環境に 配置します。Visual Studio 2022 では、Visual Studio の Marketplace より「Microsoft Visual Studio Installer Projects」をダウンロードし、インストールすることで下図のような各種インストーラー作成用のプロジェクトをご利 用できます。なお、本番環境には COBOL Server が事前にインストールされている必要があります。

拡張機能	マネージャー キ × Program1.cbl		-
参 Installe	照 インストール済 更新プログラム ローミングされ、 r project x ・ マ に ほ 巻 ・・・ Microsoft Visual Studio Installer Projects 2022 人 596.418	Microsoft Visual Studio Installer Pro	oject: ★ (52)
Ť	This official Microsoft extension provides support for Visual Studio Installer Projects in Visual Studio 2022. Microsoft インストール	インストール ブラウザーで表示 🛙 不正使用行	を報告 🖸
	Visual & Installer し 54,941 Visual Studio extension for creating NSIS and Inno Setup installers. It integrates NSIS (Nullsoft Scriptable Install Sys unSigned インストール	ARM64 NOTE: If you are running Visual Studio on an ARM64 OS use this version instead. This extension provides the same functionality that currently exists in Visual Studio 2019 for Visual Studio Installer projects. To use this extension, you can either open	

3.2 はじめての Visual COBOL プロジェクト

1) Visual Studio 2022 を起動します。

Windows のスタートメニューから Visual Studio 2022 をクリックします。Microsoft Visual Studio 2022 が起動 されるので「新しいプロジェクトの作成」を選択します。

フィルター画面が表示されるので、言語に "COBOL"、プラットフォームに "Windows"、プロジェクト タイプに "ネーティブ "を選択し、一覧から「コンソールアプリケーション」を選んで、[次へ(N)] ボタ

ンをクリックします。



2) 以下の入力を行い、[作成(C)] ボタンをクリックします。

プロジェクト名: ConsoleHello

ソリューション名: TutorialSol



ConsoleHello 新(L) C*Users*tarot*source¥repos ・	プロジェクト名(J)			
時に() C*Users¥tarot¥source¥repos ・ ・ リューションなん(M) ① TutorialSol 〕 ソリューションとプロジェクトを同じディレクトリに配置する(D) Dジェクト は *C:¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥* で作成されます	ConsoleHello			
C*Users¥tarot¥source¥repos ・ … リューション名(M) ① TutorialSol 〕 ソリューションとプロジェクトを同じディレクトリに配置する(D) ロジェクト は *C.¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥* で作成されます	易所(L)			
リューションと(M) ① TutorialSol 〕 ソリューションとプロジェクトを同じディレクトリに配置する(D) ロジェクトは *C:¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥" で作成されます	C.¥Users¥tarot¥source¥repos •			
TutorialSol 」 ソリューションとプロジェクトを同じディレクトリに配置する(D) ロジェクト は "C:¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥" で作成されます	リューション名(M) 🛈			
」 ソリューションとプロジェクトを同じディレクトリに配置する(D) ロジェクト は "C.¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥" で作成されます	TutorialSol			
	ソリューションとプロジェクトを同じディレクトリに配置する(D) 加ジェクト は "C.¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥" で作成されます			

3) Visual Studio のコードエディターで COBOL のソースコードを編集します。

ConsoleHello プロジェクトの作成が成功すると、COBOL 専用のコードエディターが起動します。エディター画面には、コ ンソールアプリケーションのひな形が表示されています。 COBOL ソースは、見出し部 (identification division)、環境 部 (environment division)、データ部 (data division)、手続き部 (procedure division) で構成されますが、 今回は「Hello World」を表示して終了するプログラムなので、手続き部に DISPLAY 文を書き加えるだけです。なお、 COBOL 正書法ではエディター画面左右にあるグレー部分を特別な領域として利用するので、通常のソースコードはこれを 避けて入力します。

今回は、procedure division の下に「Display "Hello World".」とタイプします。

Program1.cbl* -	
CBL ConsoleHello	Agent PROGRAM1 Agent Procedure Division
Y	identification division. program-id. Program1.
-	environment division. configuration section.
	data division. working-storage section.
	presedere division. Display "Hello World".
	end program Program1.

- 4) COBOL アプリケーションをビルドします。
 - 終止符(ピリオド)を含めてスペルミスがなければ、メニューより、[ファイル(F)] > [Program1.cbl の保存]を選択します。



ファイ	()l/(F)	編集(E)	表示(V)	Git(G)	プロジェクト(P)
	新規作	₣成(N)			
	開く(O)			
*	リポジ	トリのクローン	(E)		
<u>م</u>	スタート	・ウィンドウ(V	V)		
	追加([D)			
	閉じる	(C)			
	-בעע	ションを閉じ	ଟ(T)		
	Live S	hare セッショ	ンを開始		
	Live S	hare セッショ	ンに参加		
B	Progra	am1.cbl の侍	롟存(S)		Ctrl+S
	名前を	付けて Prog	ram1.cbl を	保存(A)	

② ソリューション構成が Debug、ソリューションプラットフォームが x64 であることを確認して、メニューより、[ビルド(B)]
 > [ソリューションのビルド(B)]を選択します。出力ウィンドウにビルド結果が表示されるので、ビルドが正常終了したことを確認します。

Debug ▼ x64 ▼ ソリューションのリビルド(R)							
u +							
四月							
出力元(S): ビルド 🗸 🖌 🖳 😓 🖆 😢							
13:47 でビルドが開始されました 1> ビルド開始: ブロジェクト: ConsoleHello, 構成: Debug x64							
1> ★ C:¥Users¥tarot¥source¥repos¥TutorialSol¥ConsoleHello¥Program1.cbl �R��⊕p⊕C����							
1> * Generating C:obj¥x64¥Debug¥Program1							
1> * Data: 96 Code: 517 Literals: 16 1、 coDou コンパイリー: 何二時約7日本に目標での決策の一次的 4 時							
1/ CODDEL コンパイル・I 画 正常報告 または取制の(A)語 - D 画 天政。 1) ConceleHalle - > C:#WearesWaret&euree&renewYinterialSalWConceleHalle&hin&v&&#Weahuv&ConceleHal</td><td></td></tr><tr><td>ビルド:1 正常終了または最新の状態、0 失敗、0 スキップ ==========</td><td></td></tr><tr><td>======== ビルド は 13:47 で完了し、01.033 秒 掛かりました =========</td><td></td></tr></tbody></table>							

5) COBOL アプリケーションをデバッグ実行します。

メニューより、[デバッグ(D)] > [ステップイン(I)] を選択すると、コマンドプロンプト画面が開き、デバッガーがステップ実行を 開始します。 デバッガーは手続き部の最初の COBOL 文である display 文を実行する前の状態で停止します。今回 は調べるローカル変数がないので、そのまま [ステップイン(I)] を選択し、ステップ実行を進めます。

ConsoleHello					
Console	identification division				
ř	program-id. Program1.				
+	environment division. configuration section.				
	data division. working-storage section.				
	procedure division.				
⇔ ∎	Display "Hello World". goback.				
	end program Program1.				

ステップイン後の状態. プロンプト画面は最小化されていることがあります。



Program1.cbl +	X		
ConsoleHello		✓	
Y	identification division. program-id. Program1.		
-	environment division. configuration section.		
-	data division. working-storage section.	Animator application	× + ~
⇒ ∎	procedure division. Display "Hello World". goback.	Hello World	
	end program Program1.		

コマンドプロンプト画面に "Hello World" が表示されたことを確認して、デバッグを終了します。

続いて、ウィンドウ画面のボタンを押して "Hello World" を表示する COBOL アプリケーションを作成します。

6) 作成したソリューションヘプロジェクトを追加します。

これまでに作成したソリューション中のソリューションエクスプローラーにて、ソリューションを選択した状態で、マウスの右クリック にてコンテクストメニューを表示し、[追加(D)] > [新しいプロジェクト(N)…] を選択します。



7) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に "COBOL"、プラットフォームに "Windows"、プロジェクト タイプに "デスクトッ プ"を選択し、一覧から「Windows フォームアプリケーション(.NET Framework)」を選んで、[次へ(N)] ボタンをクリ ックします。



COBOL	 ▼ Windows ▼ デスクトップ 	-
O.	WCF サービス ライプラリ (.NET Framework) WCF サービス クラス ライブラリを作成するためのプロジェクトです。	新規
	COBOL Windows デスクトップ ライブラリ	
Ő.	配信サービス ライプラリ (.NET Framework) WCF サービスとして公開される配信サービスです。	新規
	COBOL Windows デスクトップ	
CBL	Windows フォーム コントロール ライブラリ (.NET Framework) Windows フォーム アプリケーションで使用するコントロールを作成するプロジェクトです。	
	COBOL Windows デスクトップ ライブラリ	
CBL	Windows フォーム アプリケーション (.NET Framework) Windows フォーム ユーザー インタフェースのあるアプリケーションを作成するためのプロジェクト	です。
	COBOL Windows デスクトップ	
	空のプロジェクト (.NET Framework) ローカル アプリケーションを作成するための空のプロジェクトです。	
	COBOL Windows デスクトップ	
	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。	
	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。	淬ヘ(N)
	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。	次へ(N)
プロジェク	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 アト名に "WinHello" を入力し、[作成(C)] ボタンをクリックします。	次^(N)
プロジェク Windows	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 Pト名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 s フォーム アプリケーション (.NET Framework) coBol Windows テスクトッフ	次へ(N)
プロジェク Windows プロジェクトを(バ WinHello	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 アト名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。	次へ(N)
プロジェク Vindows プロジェクを&(/ WinHello 場所()	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 Pト名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 5 フォーム アプリケーション (.NET Framework) COBOL Windows デスクトップ	次へ(N)
プロジェク Windows プロジェクト名(J) WinHello 場所(L) C:¥Users¥tar	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 Pト名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 s フォーム アプリケーション (.NET Framework) COBOL Windows デスクトップ	次へ(N)
プロジェク ソロジェク Windows プロジェクト名(パ、 WinHello 場所(1) C:¥Users¥tat フレームワーク(F) .NET Framew	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 や名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 5 フォーム アプリケーション (.NET Framework) COBOL Windows デスクトップ	灾へ(N)
プロジェクト名() 「ロジェクト名() Windows プロジェクト名() WinHello 場所(L) C:¥Users¥tai フレームワーク(F, .NET Framew プロジェクトは"	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 Pト名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 s フォーム アプリケーション (.NET Framework) coBoL Windows デスクトップ mot¥source¥repos¥TutorialSol ・ …) work 4.7.2 ・ C:¥Users¥tarot¥source¥repos¥TutorialSol¥WinHello¥" で作成されます	次へ(N)
プロジェク グロジェクス Windows プロジェクト名(/ WinHello 場所(L) C¥Users¥tar フレームワーク(F) .NET Framew プロジェクトは **	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 シート名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 マフオーム アプリケーション (.NET Framework) cosol Windows デスクトップ) mort#source¥repos¥TutorialSol work 4.7.2 ・ CYUsers¥tarot¥source¥repos¥TutorialSol¥WinHello¥"で作成されます	灾へ(N)
レンション プロジェクト名(/ Windows プロジェクト名(/ WinHello 場所(L) C:¥Users¥tal フレームワーク(F) .NET Framew プロジェクトは、************************************	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 やわ名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 マフォーム アプリケーション (.NET Framework) cosol Windows デスクトップ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	次へ(N)
プロジェク Vindows プロジェクをな() WinHello 場所(L) C¥Users¥tal フレームワーク(F) .NET Framew プロジェクトは **	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 シート名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 マフォーム アプリケーション (.NET Framework) COBOL Windows デスクトップ (次へ(N)
レンション プロジェクト名(/ Windows プロジェクト名(/ WinHello 場所(L) C:¥Users¥tal フレームワーク(F) .NET Framew プロジェクトは	ユニット テスト ライブラリ (NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 やわ名に"WinHello"を入力し、[作成(C)] ボタンをクリックします。 マオーム アプリケーション (.NET Framework) cosol Windows デスクトップ ・ rot¥source¥repos¥TutorialSol ・ … ・ cvt4.7.2 ・ C¥Users¥tarot¥source¥repos¥TutorialSol¥WinHello¥"で作成されます	次へ(N)
プロジェク Vindows プロジェクをな() WinHello 場所(L) C¥Users¥tar フレームワーク(F) .NET Framew プロジェクトは**	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 やわ名に"WinHello"を入力し、[作成(C)] ボタンをクリックします。 マオーム アプリケーション (.NET Framework) cool Windows デスクトップ (茨へ(N)
レンション プロジェク びindows プロジェクト名(パ) WinHello 場所(1) C:¥Users¥tar フレームワーク(F) .NET Framev プロジェクトは、************************************	ユニット テスト ライブラリを作成するための .NET プロジェクトです。 かわ名に "WinHello"を入力し、[作成(C)] ボタンをクリックします。 マカーム アプリケーション (.NET Framework) cool Windows テスクトップ 	次へ(N)
レンション プロジェクト名() Windows プロジェクト名() WinHello 場所(1) C:¥Users¥tat フレームワーク(F) NET Framev プロジェクトは・	ユニット テスト ライブラリ (.NET Framework) MFUnit テスト ライブラリを作成するための .NET プロジェクトです。 やち名に "WinHello" を入力し、[作成(C)] ボタンをクリックします。 * フォーム アプリケーション (.NET Framework) cosol Windows デスクトップ ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	<u>次</u> ∧(N)

WinHello プロジェクトの作成が成功すると、フォームデザイナーが起動します。[表示] メニューから[ツールボックス] を選 択しておきます。

9



デザイナー画面に Form1 ウィンドウが表示されるので [表示(V)] > [ツールボックス(X)] を選択して展開します。表示されたツールボックス中のすべての Windows フォームを展開します。続いて、Button コントロールを選択し、Form1 ウィンドウ上にドラッグ&ドロップします。



Form1 ウィンドウ上にボタンが表示されると、プロパティが Button1 ボタンに切り替わります。プロパティを下方向にスクロ ールして「Text」を選択し、値を "Say Hello" に変更します。

プロパティ	~ ₽
button2 System.Windows.Fo	rms.Button
🔡 🐏 🕐 🔑	
Image	(なし)
ImageAlign	MiddleCenter
ImageIndex	(なし)
ImageKey	(なし)
ImageList	(なし)
RightToLeft	No
Text	button1
TextAlign	MiddleCenter

ツールボックスをスクロールして Label コントロールを選択し、Form1 ウィンドウ上にドラッグ&ドロップします。



プロパティをスクロールして「Text」を選択し、テキストの値を削除します。



プ	コパティ	▼ ‡	×
la	bel1 System.Windows.Forms	.Label	•
0	🗄 💱 🖗 🦻 🦻		
	BorderStyle	None	*
	Cursor	Default	
	FlatStyle	Standard	
Ð	Font	MS UI Gothic, 7.20000029pt	
	ForeColor	ControlText	
	Image	(なし)	
	ImageAlign	MiddleCenter	
	ImageIndex	(なし)	
	ImageKey	(なし)	н
	ImageList	(なし)	
	RightToLeft	No	
	Text	×	-

10) コードエディターで COBOL ソースコードを入力します。

デザイナー画面上の [Say Hello] ボタンをダブルクリックすると、COBOL 専用のコードエディターが起動します。 エディター画面には、Windows フォームアプリケーションのひな形が表示されます。 ここでは [Say Hello] ボタンをクリッ クした時の処理を記述するので、 Button1_Click メソッドの手続き部に以下の move 文を追加します。

move "Hello World!" to self::label1::Text.

Form1.cbl* 🚽 🗙 Form1.cbl [デザイン]*	Program1.cbl
CBL WinHello	✓ ♥ WinHello.Form1 ✓
✓ procedure division. invoke self::Initiali goback. end method. 0個の参照 method-id button1_Click f > procedure division using move "Hello World!" t end method.	zeComponent inal private. by value sender as object e as type System.EventArgs. to self::label1::Text.
end class.	

11) COBOL アプリケーションをビルドします。

メニューより、[ファイル(F)] > [Form1.cbl の保存] を選択し、変更を保存します。

ファイ	´ル(F)	編集(E)	表示(V)	Git(G)	プロジェクト(P)				
	新規作	乍成(N)							
	開く(O)								
4	リポジトリのクローン(E)								
<mark>۵</mark> -] スタート ウィンドウ(W)								
	追加(D)								
	閉じる	(C)							
×	-בעע	・ションを閉じ	ଟ(T)						
	Live S	hare セッショ	ンを開始						
	Live S	hare セッショ	ンに参加…						
B	Form	I.cbl の保存((S)		Ctrl+S				

メニューより、[ビルド(B)] > [WinHello のビルド(U)] を選択します。



ビルド(B) デバッグ(D) テスト(S) 分析(N) ツール(T)
 ソリューションのビルド(B) Ctrl+Shift+B ソリューションのリビルド(R) ソリューションのクリーン(C) ソリューションでコード分析を実行(Y) Alt+F11
 WinHelloのビルド(U) Ctrl+B

出力ウィンドウにビルド結果が表示されますので、ビルドが正常終了したことを確認します.

出力
出力元(S): ビルド 🔹 🖕 🕒 🙂
1> ★ C:¥Users¥tarot¥source¥repos¥TutorialSol¥WinHello¥Main.cbl ∲R♦♦♦₽♦C♦♦♦₽
1> * 000000 WinHello.Form1
1> ★ ♦♦♦♦₽♦ WinHello.Main
1> WinHello -> C:¥Users¥tarot¥source¥repos¥TutorialSol¥WinHello¥bin¥Debug¥WinHello.exe
ビルド は 14:21 で完了し、00.733 秒 掛かりました ========

12) COBOL アプリケーションを実行します。

ソリューションエクスプローラーにて WinHello プロジェクトを選択した状態で、マウスの右クリックにてコンテクストメニューを表

示し、[スタートアッププロジェクトに設定(A)] を選択します。

ソリューション エクスプローラー								
ソリューション エクスプロー	ラーの	検索 (Ctrl+:)						
Vリューション 'Tute ConsoleHell Propertie Program	*	ビルド(U) リビルド(E) クリーン(N)						
▲ III WinHello ✓ Propertie ▶ ►□ 参照設定	•	すべての子孫を折りたたむ ここまでスコープ指定する(S) 新しいソリューション エクスプローラーのビュー(N)	Ctrl+左矢印					
Form1.cb アリューション エクスプロー		ビルドの依存関係(B) 追加(D) 既存の COBOL 項目を追加	> >					
WinHello プロジェクト	•	NuGet パッケージの管理(N) スタートアッププロジェクトの壊成						
日 その他		スタートアップ プロジェクトに設定(A)						

メニューより、[デバッグ(D)] > [デバッグなしで開始(H)] を選択すると、Form1 ウィンドウが開きます。

デバ	ッグ(D)	テスト(S)	分析(N)	ツール(T)	拡張機能(X)
	ウィンド	ל(W)			
	デバッグ	の開始(S)			F5
\triangleright	デバッグ	なしで開始(ト	H)		Ctrl+F5
2	パフォー	マンス プロファ	イラー(F)		Alt+F2

Form1 ウィンドウの [Say Hello] ボタンをクリックすると "Hello World!" が表示されます。



🖳 Form1		-	×
	Hello World!		
	Say Hello		

アプリケーションの終了は、[X] ボタンをクリックします。

3.3 ファイルの入出力

続いて、エクセルやメモ帳で作成した CSV ファイルを読み込んで、固定長順編成ファイルを作成する COBOL アプリケーション を作成します。

1) 作成したソリューションヘプロジェクトを追加します。

さきほどまで使用していたソリューション中のソリューションエクスプローラーにて、ソリューションを選択した状態で、マウスの右ク リックにてコンテクストメニューを表示し、[追加(D)] > [新しいプロジェクト(N)...] を選択します。

→ ☆	ソリューション エクスプロ-	-ラ-		- ₽ ×
×		þ	» 🔁	
	ソリューション エクスプロ	-ラ-	の検索 (Ctrl+:)	- م
	 □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□	*	ソリューションのビルド(B) ソリューションのリビルド(B)	Ctrl+Shift+B
	▷ Image: WinHello		ソリューションのクリーン(C)	
			分析とコードのクリーンアップ(A)	•
			バッチ ビルド(T)…	
			構成マネージャー(O)	
			ソリューションの NuGet パッケージの管理(N)…	
	ソリューション エクスプロ	[2	NuGet パッケージの復元(G)	
	プロパティ	Ξ	すべての子孫を折りたたむ	Ctrl+左矢印
	TutorialSol ソリューシ	<u>*</u> =	新しいソリューション エクスプローラーのビュー(N)	
	🔡 💱 🏓		プロジェクトの依存関係(S)	
	日 その他		プロジェクトのビルド順序(I)	
新しいプロジェクト(N)			追加(D)	•
既存のプロジェクト(E)		£Ç;}	スタートアップ プロジェクトの構成	

2) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に "COBOL"、プラットフォームに "Windows"、プロジェクト タイプに "ネーティブ "を選択し、一覧から「コンソールアプリケーション」を選んで、[次へ(N)] ボタンをクリックします。



3)

COBOL	 Windows オーティブ
	Windows アプリケーション Windows アプリケーションを作成するためのネイティブ プロジェクトです。 COBOL Windows ネーティブ
CBL C1	コンソール アプリケーション ネイティブ コマンドライン アプリケーションを作成するためのプロジェクトです。 COBOL Windows ネーティブ コンソール
	空のプロジェクト ローカル アプリケーションを作成するための空のプロジェクトです。 COBOL Windows ネーティブ
	リンク ライブラリ 他のアプリケーションで使用するクラスを作成するためのネイティブ プロジェクトです。 COBOL Windows ネーティブ ライブラリ
	ユニット テスト ライブラリ MFUnit テスト ライブラリを作成するためのネイティブ プロジェクトです。 COBOL Windows ネーティブ テスト ライプラリ
CBL	Micro Focus INT/GNT Micro Focus INT または GNT コードを作成するためのプロジェクトです。
	次へ(N)

プロジェクト名に "LoadCSVFile" を入力し、[作成(C)] ボタンをクリックします。

コンソール アプリケーション COBOL Windows ネーティブ コンソール	
加ジェクト名()	
LoadCSVFile	
場所(L)	
C.¥Users¥tarot¥source¥repos¥TutorialSol	
	戻る(B) 作
]ードエディターで COBOL ソースコードを入力します。	

LoadCSVFile プロジェクトの作成が成功すると、COBOL 専用のコードエディターが起動します。エディター画面にコンソ ールアプリケーションのひな形が表示されるので、環境部 (environment division)、データ部 (data division)、手続 き部(procedure division) を書き換えます。

まず、環境部の構成節 (configuration section) を削除し、以下の入出力節 (input-output section) を追加し



ます。まだ、データ部のファイル定義が未入力なので IN-FILE と OUT-FILE がエラーとなりますが、ここでは無視して構いません。

INF	PUT-OUTPUT S	ECTION.				
FIL	E-CONTROL.					
	SELECT IN-F	ILE ASSIGN	TO "Emp M	laster.	CSV"	
			····			
			TO "F			
	SELECT OUT	-FILE ASSIGN	TO "Emp_I	Maste	r.dat".	
Program1.cbl*	≄ × Form1.cbl	Form1.cbl [デザイン]	Program1.cbl			~ *
CBL LoadCSVFile	ident if i entire divici	✓ ♥ PROGRAM1	•	E Worki	ng-Storage Section	
ř	program-id. Program1.	on.				alaan aa tu
Ň	environment division. INPUT-OUTPUT SECTION.					
	SELECT IN-EILE	ASSIGN TO "Emp_Maste	r.csv″			
-	SELECT QUI-FILE	ASSIGN TO "Emp_Maste	r.dat".			17
Ĭ	data division. working-storage secti	on.				-
Ţ	procedure division					
	goback.					
	end program Program1.					
						_
100 % 🔻 🔇	2 ▲0 ↑ ↓	4			▶ 行: 10 文字: 1	SPC CRLF
Iラ-一覧						- ₽ ×
ソリューション全体	- 🛛 🕹 2 IJ-	🛕 0 警告 🚺 0 メッセー	ジ 💡 ビルド + Intelli	Sense	 ▼ エラー一覧を検索 	- 9
۲-۴	説明	プロジェクト	ファイル	行	プログラム	連続
😣 совсн	10, ファイル IN-FILE に対する FD 記述項がない	LoadCSVFile	Program1.cbl	7	Program1.cbl	7
😣 совсн	10, ファイル OUT-FILE に対する FD 記述項がない	LoadCSVFile	Program1.cbl	9	Program1.cbl	9

次に、データ部の作業場所節 (working-storage section) を削除し、以下のファイル節 (file section) を追加しま す。なお、データ部のファイル定義を入力したので、環境部のエラーは無くなります。

FI	_E SE	ECTION.	
FD	IN.	-FILE.	
01	IN-	REC	PIC X(50).
FD	OU	T-FILE.	
01	OU	T-REC.	
	05	OUT-EMPNO	PIC 9(8).
	05	FILLER	PIC X.
	05	OUT-JNAME1	PIC X(10).
	05	OUT-JNAME2	PIC X(10).
	05	OUT-NAME1	PIC X(5).
	05	OUT-NAME2	PIC X(5).
	05	OUT-GENDER	PIC X.
	05	FILLER	PIC X.
	05	OUT-DIV	PIC X(10).
	05	OUT-EMPDATE	PIC 9(8).
	05	FILLER	PIC X.





最後に、手続き部の goback 文を削除し、以下の 手続き文を追加します。

```
PROC1.
   OPEN INPUT IN-FILE.
   OPEN OUTPUT OUT-FILE.
PROC2.
   READ IN-FILE AT END
                         GO TO PROC9.
   INITIALIZE OUT-REC.
   UNSTRING IN-REC DELIMITED BY ","
    INTO OUT-EMPNO
         OUT-JNAME1
         OUT-JNAME2
         OUT-NAME1
         OUT-NAME2
         OUT-GENDER
         OUT-DIV
         OUT-EMPDATE
   END-UNSTRING.
   WRITE OUT-REC.
   GO TO PROC2.
PROC9.
   CLOSE IN-FILE OUT-FILE.
   STOP RUN.
```





4) COBOL アプリケーションをビルドします。

メニューより、[ビルド(B)] > [LoadCSVFile のビルド(U)] を選択します。

ビル	۴̈(B)	デバッグ(D)	テスト(S)	分析(N) ツール(T)		
ゾリューションのビルド(B) Ctrl+Shift+							
	בעצ	ーションのリビル	ŀ°(R)				
	ソリューションのクリーン(C)						
	בעצ	Alt+F11					
÷.	Load	ICSVFile のビル	۴(U)		Ctrl+B		
	Load	ICSVFile のリヒ	レド(E)				

出力ウィンドウにビルド結果が表示されますので、ビルドが正常終了したことを確認します.

出力
出力元(S): ビルド 🔹 🚽 🖳 😓 🗮 🛤 🕑
1> * C:¥Users¥tarot¥source¥repos¥TutorialSol¥LoadCSVFile¥Program1.cbl ♥R♦♦♦p♦C♦♦♦ 1> * Generating C:obj¥x64¥Debug¥Program1 1> * Data: 1216 Code: 1851 Literals: 48 1> COBOL コンパイル:1 個 正常終了または最新の状態 0 個 失敗。 1> LoadCSVFile -> C:¥Users¥tarot¥source¥repos¥TutorialSol¥LoadCSVFile¥bin¥x64¥Debug¥LoadCSVFile.exe ===================================
=====================================

5) CSV ファイルを作成します。

デバッグフォルダ(<3.2 節の 3) 「場所」で指定したフォルダ> ¥TutorialSol¥LoadCSVFile¥bin¥x64¥debug)

にメモ帳などを利用して以下の Emp_Master.csv ファイルを作成します。

11111113,佐藤,隆,サトウ,タカシ,M,営業部,19980401,0
22222226,鈴木,尚之,スズキ,ナオユキ,M,技術部,19981015,0
33333339,田中,直美,タナカ,ナオミ,F,総務部,19990401,0
44444442,山田,洋一,ヤマダ,ヨウイチ,M,営業部,20000701,0
55555555,伊藤,弘子,仆ウ,ヒロコ,F,技術部,20010401,0
66666668,木村,貴弘,キムラ,タカヒロ,M,営業部,20021220,0
77777771,中村,慎司,ナカムラ,シンジ,M,技術部,20030401,0
88888884,橋本,悦子,心モト,エツコ,F,総務部,20040805,0



99999997,三井,薫,ミツイ,カオル,F,営業部,20050401,0

注意)

上記ファイルは Shift_JIS 形式で保存してください。メモ帳を使用する場合、Windows 製品のバージョンによって保存 する文字コード形式が異なりますので、ANSI を選択してください。

名前	^
Emp_Master.csv	
LoadCSVFile.exe	
_	

6) COBOL アプリケーションをデバッグ実行します。

Program1.idy

ソリューションエクスプローラーにて LoadCSVFile プロジェクトを選択した状態で、マウスの右クリックにてコンテクストメニュー を表示し、[スタートアッププロジェクトに設定(A)] を選択します。



続いて、メニューより、[デバッグ(D)] > [ステップイン(I)] を選択して、デバッガーによるステップ実行を開始します。 デバッ ガーは手続き部の最初の COBOL 文である open 文で実行を中断します。



入力ファイルから読み込んだレコードの内容を確認するため、unstring 文の in-rec 上でマウスの右クリックにてコンテクス トメニューを表示し、[ウォッチの追加(W)]を選択します。



同様に出力ファイルに書き出すレコードの内容を確認するため、initialize 文の out-rec 上でもマウスの右クリックにてコン テクストメニューを表示し、[ウォッチの追加(W)] を選択します。



F11 キーを 3 回押すと、デバッガーは read 文実行後、処理を中断します。ウォッチ式の in-rec の値には CSV ファ イルから読み込んだ最初のレコードが表示されます。





さらに F11 キーを 2 回押すと、デバッガーは unstring 文を実行後、処理を中断します。ウォッチ式の out-rec の値 には出力ファイルへ書き出す最初のレコードが表示されます。



Program1.cbl	÷Þ	х	Form1.cbl	Form1.cbl	[デザイン]	Program	m1.cbl			
LoadCSVFile	e					-	PROGRAM	//1		
¢	F	1個 2R00	INITIALIZE OU UNSTRING IN-RI INTO OUT-JNAME1 OUT-JNAME2 OUT-NAME1 OUT-OENDER OUT-GENDER OUT-GENDER OUT-EMPDATE END-UNSTRING. WRITE OUT-REC GO TO PROC2. の参照 23. CLOSE IN-FILE STOP RUN. program Program	OUT-FILE.	D BY ","					
100 % 🝷	0	問題	は見つかりませんでし	たく					_	
ウォッチ 1										- q
検索 (Ctrl+E)			P	\bullet \leftrightarrow \rightarrow to \bullet	検索の詳細度:	-				
名前				値					種類	
🤗 IN-REC	,			11111113,4	左藤,隆,サトウ,タカ	シ,M,営業部,	,19980401 Q	表示 🔻	PIC X(50)	
▶ 🔗 OUT-R	EC	-40	+7	{長さ=60}:	"11111113 佐	藤隆	ታኑኃ ዓክ… 🔍	表示 ▼	GROUP	
「現日をリオツナ	に足	2/)[[90							

さらに F11 キーを 4 回押すと、デバッガーは initialize 文を実行後、処理を中断します。ウォッチ式の in-rec の値に は CSV ファイルから読み込んだ 2 番目のレコードが表示され、out-rec の値は initialize 文で初期化されています。





メニューより、[デバッグ(D)] > [続行(C)] を選択するか、CSV ファイルからすべてのレコードを読み込むまで F11 キーを 押すと、デバッガーは終了します。

出力	▼ ₽ ×
出力元(5): 「パッグ ・ 〜 〜 〜 竺 二 二 二 四 〇	
スレッド '(名前がありません)' (9712) はコード 0 (0x0) で終了しました。 スレッド '(名前がありません)' (9148) はコード 0 (0x0) で終了しました。 スレッド '(名前がありません)' (1936) はコード 0 (0x0) で終了しました。 スレッド '(名前がありません)' (1922) はコード 0 (0x0) で終了しました。 スレッド '(名前がありません)' (1922) はコード 0 (0x0) で終了しました。 プログラム '[8524] LoadCSVFile.exe: c:¥users¥tarot¥source¥repos¥tutoria!sol¥loadcsvfile¥bin¥x64¥debug¥LoadCSVFile.exe' はコード 0 (0x0) で終了しました。	
4	
15	

さきほど Emp_Master.csv を作成したフォルダ配下に Emp_Master.dat ファイルが作成されます。テキストエディタなど でファイルを開き、社員 9 名分のデータが表示されることを確認します。エディター上で 60 桁で折り返し設定を行うと、以 下のように表示されます。



0	. 1		. 3		. 4	4	5
11111113	佐藤	隆	ታኮታ	タカシ	Μ	営業部	19980401
22222226	鈴木	尚之	77.*+	ナオユキ	M	技術部	19981015
33333339	田中	直美	タナカ	ナオミ	F	総務部	19990401
44444442	Ш 🖽	洋一	779°	ヨウイチ	M	営業部	20000701
55555555	伊藤	弘子	ለኮታ	երյ	F	技術部	20010401
66666668	木村	貴弘	Ŧ45	タカヒロ	M	営業部	20021220
7777771	中村	慎司	ナカムラ	シンシ゛	M	技術部	20030401
88888884	橋本	悦子	ハシモト	エツコ	F	総務部	20040805
999999997	三井	薫	ミツイ	カオル	F	営業部	20050401

さきほどの固定長順編成ファイルを読み込んでレポートファイルを作成するバッチアプリケーションを作成します。

7) ソリューションに新規プロジェクトを追加します。

ソリューションエクスプローラーにて、TutorialSol ソリューションを選択した状態で、マウスの右クリックにてコンテクストメニュー を表示し、[追加(D)] > [新しいプロジェクト(N)...]を選択します。



8) 使用するテンプレートを選択します。

フィルター画面が表示されるので、言語に "COBOL"、プラットフォームに "Windows"、プロジェクト タイプに "ネーティブ "を選択し、一覧から「コンソールアプリケーション」を選んで、[次へ(N)] ボタンをクリックします。



COBOL	 Windows ネーティブ 	j -
	Windows アプリケーション Windows アプリケーションを作成するためのネイティブ プロジェクトです。 COBOL Windows ネーティブ	
C:\	コンソール アプリケーション ネイティブ コマンドライン アプリケーションを作成するためのプロジェクトです。 COBOL Windows ネーティブ コンソール	
	空のプロジェクト ローカル アプリケーションを作成するための空のプロジェクトです。 COBOL Windows ネーティブ	
	リンク ライブラリ 他のアプリケーションで使用するクラスを作成するためのネイティブ プロジェクトです。	2
- (1)	COBOL Windows ネーティブ ライブラリ	
۶.	ユニット テスト ライフラリ MFUnit テスト ライブラリを作成するためのネイティブ プロジェクトです。	
	COBOL Windows ネーティブ テスト ライブラリ	
CBL	Micro Focus INT/GNT Micro Focus INT または GNT コードを作成するためのプロジェクトです。	
		次へ(N)

9) プロジェクト名に "BATCHRPT" を入力し、[作成(C)] ボタンをクリックします。

シェクト名()
ATCHRPT
f(L)
¥Users¥tarot¥source¥repos¥TutorialSol •
戻る(B) 作

BATCHRPT プロジェクトの作成が成功すると、COBOL 専用のコードエディターが起動します。エディター画面にコンソー



ルアプリケーションのひな形が表示されるので、ソリューションエクスプローラーで BATCHRPT プロジェクト配下のソースプロ グラム「Program1.cbl」を選択した上で、マウスの右クリックにてコンテクストメニューを表示し、[名前の変更(M)] を選 択します。その後、プログラム名を "BATCHRPT.cbl" に書き換えます。

ソリューション エクスプローラー			▼ ₽					
√ [™] → → → → / × → / ×	Ŧ							
ソリューション エクスプローラー の検索 (Ctrl+:)								
🔀 ソリューション 'TutorialSol' (4/4 の)	プロジュ	27F)						
▲ CBL BATCHRPT								
🎤 Properties								
Program1.cbl	\rightarrow	聞<(○)						
ConsoleHello		コッイルを関ノマゴルケーションの避力(N)						
LoadCSVFile		ノアイルを用くアフリケーションの選び(IN)						
▷ I WinHello		コードのクリーンアップ	•					
	ĩ	コンパイル(M)	Ctrl+F7					
ソリューション エクスプローラー Git 変更		ここまでスコープ指定する(S)						
プロパティ	<u>*</u> =	新しいソリューション エクスプローラーのビュー(N)						
Program1.cbl コンパイル項目のプロパテ		比較する(W)						
		プロジェクトから除外())						
□ その他								
ファイル名		COBOLプロジェクトを作成						
完全パス	X	切り取り(T)	Ctrl+X					
□ 高度	б	コピー(Y)	Ctrl+C					
カスタム ツール			Del					
カスタムツールの名前空間	Â		Dei					
コマンドライン	Ē	名前の変更(M)	F2					

変更後のソリューションエクスプローラー

ソリューション エクスプローラー
J O - ≒ □ @ / M i ▼
ソリューション エクスプローラー の検索 (Ctrl+:)
🔀 ソリューション 'TutorialSol' (4/4 のプロジェクト)
BATCHRPT
🎤 Properties
BATCHRPT.cbl
▷ ConsoleHello
LoadCSVFile
▶ 💷 WinHello

サンプルプログラム BARCHRPT.cbl の内容でコードを上書き保存します。



BATCHRPT.cbl	😐 🗙 Program1.cbl	Form1.cb	I Form1.cbl [デサ	イン] Pi	rogram1.cbl		< - ¢
BATCHRPT		 Agent Batchrpt 	Г		KIT()	-	÷
> >	IDENTIFICATION C PROGRAM-ID. BATC ************************************	DIVISION. HRPT. cocesses files: s = Employee E Selection e = Employee Y coccesses	******************** xtract File (Sequent Control Card rs Employed Report **********	************ * ial) * * * *			A TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
->->	ENVIRONMENT DIVI INPUT-OUTPUT SEC FILE-CONTROL. * INPUT FILE: EMP	SION. TION.					Mile 2 Marcel Mile 2 Marcel Mi
	SELECT EMP-S * INPUT FILE: DAT SELECT IN-CN * OUTPUT REPORT F SELECT EMP-F	EU-FILE A E SELECTION CRI ITL-CARD A ILE URE-RPT A	SSIGN TO UT-S-EMPSEQ TERIA SSIGN TO UT-S-CNTLCA SSIGN TO UT-S-HIRERP	RD. т			Anno Annother Annothe
>·> 	DATA DIVISION. FILE SECTION.						
100 % 🔹		\downarrow \triangleleft			▶ 行: 260	文字: 29 9	SPC CRLF
Iラー一覧							- 4 ×
ソリューション全	体 👻 🛛) エラー 🛕 0 警領	青 🚺 0 メッセージ 🛛 💡	ビルド + Inte	IliSense 👻 🛛	Lラー一覧を核	索 👂 -
[™] コード	説明	プロジェクト		1	行 プログラム		
😢 сово	コピーブック EMPSEC が見つからない	BATCHRPT	BATCHRPT.cb	I (52 BATCHRPT.cb	I	
😵 сово	このレベルの前述項 CHO2目の長さがゼロであ る	BATCHRPT	BATCHRPT.cb	1 (54 BATCHRPT.cb	1	
1ラ-一覧 出力	נ						

参照しているコピーブックが存在しないため、エラーが報告されますが、ここでは無視して構いません。 続いて、参照されるコピーブックを作成します。

TutorialSol ソリューション配下の BARCHRPT プロジェクト名を選択した上で、マウスの右クリックにてコンテクストメニュ ーを表示し、[追加(D)] > [新しい項目(W)] を選択します。



▲ インストール済み		並べ替え	: 既定	• III 😑		検索 (Ctrl+E)	P
 COBOL プロジェクト Code 	项目	٥	COBOL プログラム		COBOL プロジェクト項目	種類: COBOL プロジェクト項目	
General Test		Z	テスト プログラム		COBOL プロジェクト項目		
▶ オンライン		D	コピーブック		COBOL プロジェクト項目		
		D	アプリケーション 構成 ファイル		COBOL プロジェクト項目		
		D	リソース ファイル		COBOL プロジェクト項目		
		D	アプリケーション マニフェスト		COBOL プロジェクト項目		
名前(N):	EMPSEQ.cpy						
						追加(A) キャ	ッンセル

「コピーブック」を選択し、名前に "EMPSEQ.cpy" を入力し、[追加(A)] ボタンをクリックします。

作成された EMPSEQ.cpy の中身を、サンプルプログラム EMPSEQ.cpy で上書き保存します。

EMPSEQ.cpy	🔁 🗙 BATCHRPT.cbl	Program1.cbl	Form1.cbl	Form1.cbl
BATCHRPT		 Comparison Comparison	•	Procedure 🍄
	* * 25 EMP-REC. 10 EMPREC-SSN P1 10 FILLER PIC X0 10 EMPREC-JNAME1 10 EMPREC-JNAME2 10 EMPREC-NAME1 10 EMPREC-NAME2 10 EMPREC-ONDEF 10 FILLER PIC X0 10 EMPREC-DIV P1 10 EMPREC-DIV	ENTIAL FILE LAYOUT ENTIAL FILE LAYOUT (01) VALUE SPACE. (01) VALUE SPACE. PIC N(05) VALUE SPACE. PIC X(05) VALUE SPACE. PIC X(05) VALUE SPACE. (01) VALUE SPACE.	EROES. DES. DES.	** * *

メニューより、[ビルド(B)] > [BATCHRPT のビルド(U)] を選択します。

ビル	/ド(B)	デバッグ(D)	テスト(S)	分析(N)	ツール(T)	
ソリューションのビルド(B) Ctrl+Shift+B						
	בעע	ーションのリビル	ŀ°(R)			
	בעצ	リューションのクリーン(C)				
	בעע	ーションでコード会	分析を実行('	Y) Al	t+F11	
i.	BATC	CHRPT のビルド	(U)	Ct	rl+B	



さきほどまでのエラーが全て解消されていることを確認します。

出力	
出力元(S): ビルド	
 * C:¥Users¥tarot¥source¥repos¥TutorialSi * Generating C:obj¥x64¥Debug¥BATCHRPT * Data: 2416 Code: 421 COBOL コンパイル:1 個 正常終了または最 BATCHRPT -> C:¥Users¥tarot¥source¥repos ======== ビルド:1 正常終了または最新の状 ======= ビルド は 15:21 で完了し、00.431 	, mp. bood act of¥BATCHRPT¥BATCHRPT.cbl ∲R♦♦♦p♦C♦♦♦♦ 56 Literals: 224 新の状態 0 個 失敗。 ¥TutorialSol¥BATCHRPT¥bin¥x64¥Debug¥BATCHRPT.exe 態、0 失敗、0 スキップ ======== 9 秒 掛かりました ==========
ソリューション エクスプローラー	
▼ 5 - 5 0 0 0	
ソリューション エクスプローラー の検索 (Ctrl+:)	
😡 ソリューション 'TutorialSol' (4/4 のプロジェクト)	
A DEL BATCHRPT	
🎤 Properties	
BATCHRPT.cbl	
EMPSEQ.cpy	
ConsoleHello	
LoadCSVFile	
▶ I WinHello	
ソリューション エクスプローラー Git 変更	

11) COBOL コンパイル指令を追加します。

ファイル名の割り当てを EXTERNAL (外部割り当て) に変更するため、ソリューションエクスプローラーにて BATCHPRT プロジェクト配下の Properties を選択し、マウスの右クリックにてコンテクストメニューを表示し、[開く(O)] を選択します。

ソリューション エクスプローラー				
₰ ७ • ≒ 🗇 🖗 🗡 🛋 🔻				
ソリューション エクスプローラー の検索 (Ctrl+:)				
□ フリューション 'TutorialSol' (4/4 のプロジェクト)				
▲ CBL BATCHRPT				
🎤 Properties				
BATCHRPT.cbl	<u>(</u> 開く(U)			
EMPSEQ.cpy	追加(D)			

BATCHRPT 🗢 🗙 EMPSEQ.cpy	BATCHRPT.cbl	Program1.cbl	Form1.cbl	
アプリケーション SQL	構成(C): アクティブな (Debu ブラットフォーム(M): アクティブ	ig) ~	<u></u>	
デバッグ				
コビーブック プリプロヤッサ	出カパス:	.¥bin¥x64¥Debug	¥	
COBOL	□ 指令ファイルの生成		🗌 IJストファイ	ルを生成
Micro Focus Code Analysis	🗌 コード カバレッジを有す	効にする	🗌 วีบวァイラ	を有効にする
	追加指令 —			
	ASSIGN(EXTERNAL)			

COBOL タブを選択し 追加指令に "ASSIGN(EXTERNAL)" を入力し、変更を保存します。

Ŧ



12) アプリケーション構成ファイルを作成します。

TutorialSol ソリューション配下の BARCHRPT プロジェクト名を選択し、マウスの右クリックにてコンテクストメニューを表示し、[追加(D)] > [新しい項目(W)] を選択します。

~ ₹ \$	<u>Vリュ-ション Iクスカーラ-</u> □ ○ · ≒ □ □ <i>▲</i> <u>V</u> リュ-ション Iクスプローラ- の枝	*	ビルド(U) リビルド(E) クリーン(N)	
	Organization A State A State	Ē	すべての子孫を折りたたむ ここまでスコープ指定する(S) 新しいソリューション エクスプローラーのビュー(N)	Ctrl+左矢印
を生成	EMPSEQ.cpy		ビルドの依存関係(B)	•
*1 新しい項目(W)	Ctrl+Shift+A		追加(D)	•

13) 使用するテンプレートを選択します。

「アプリケーション構成ファイル」を選択し、[追加(A)]をクリックします。

▲ インストール済み	並べれ	春え: 既定	• III 😑		検索 (Ctrl+E)		ہ
▲ COBOL プロジェクト項目 Code	D	COBOL プログラム		COBOL プロジェクト項目	種類: COBOL プロジョ	[クト項目	-
General Test	Ľ	テストプログラム		COBOL プロジェクト項目	アノリリーションの設定す イルです。	と悔成するために1史り。	131
▶ オンライン	Ē	コピーブック		COBOL プロジェクト項目			
	Ļ	アプリケーション 構成 ファイル		COBOL プロジェクト項目			
	Ē	リソース ファイル		COBOL プロジェクト項目			
	Į.	アプリケーション マニフェスト		COBOL プロジェクト項目			
名前(N): App	lication1.mfgcf						
					追加	I(A) キャンセル	

生成された Application1.mfgcf ファイルをダブルクリックし、表示されたダイアログで [追加] をクリックし、以下の入力 を行います。

変数	值			
dd_EMPSEQ	Emp_Master.dat			
dd_CNTLCARD	Cntl_Card.dat			
dd_HIRERPT	Hire_Report.dat			



dd_EMPSEQ	Emp_Master.dat)é ±n
dd_CNTLCARD	Cntl_Card.dat	2巨川1
dd_HIRERPT	Hire_Report.dat	削除
	dd_EMPSEQ dd_CNTLCARD dd_HIRERPT	dd_EMPSEQ Emp_Master.dat dd_CNTLCARD Cntl_Card.dat dd_HIRERPT Hire_Report.dat

上記のようになっていることを確認のうえ、[OK] をクリックします。

14) COBOL アプリケーションをビルドします。

メニューより、[ビルド(B)] > [BATCHRPT のリビルド(E)] を選択します。

ビル	ኑ ["] (B)	デバッグ(D)	テスト(S)	分析(N)	ツール(T)		
*	ソリューションのビルド(B) Ctrl+Shift+B						
	ソリューションのリビルド(R)						
	ソリューションのクリーン(C)						
	ソリューションでコード分析を実行(Y) Alt-				t+F11		
	BATCHRPT のビルド(U) Ctrl+B				rl+B		
	BATC	CHRPT のリビル	ド(E)				
	BATO	CHRPT のクリーン	ン(N)				

出力ウィンドウにビルド結果が表示されるので、すべてのビルドが正常終了したことを確認します。





15) 入力ファイルをコピーします。

手順 6)の最後で作成された Emp_Master.dat ファイルをデバッグフォルダ (<3.2 節の 3) 「場所」で指定したフォ ルダ> ¥TutorialSol¥BATCHRPT¥bin¥x64¥debug) にコピーします。

- 名前
- BATCHRPT.exe
- BATCHRPT.exe.mfgcf
- BATCHRPT.idy
- Emp_Master.dat
- 16)制御ファイルを作成します。

"20000101" が記載された Cntl_Card.dat ファイルをデバッグフォルダ (<3.2 節の 3) 「場所」で指定したフォルダ> ¥TutorialSol¥BATCHRPT¥bin¥x64¥debug) に作成します。

名前	名前 更新日時	
BATCHRPT.exe	2024/08/26 15:37	<u>ም</u> ታህታ-
BATCHRPT.exe.mfgcf	Cntl_Card.dat	×
BATCHRPT.idy	ファイル 編集 表示	
Cntl_Card.dat	////// 顺未 	
Emp_Master.dat	20000101	

17) COBOL アプリケーションをデバッグ実行します。

ソリューションエクスプローラーにて BATCHRPT プロジェクトを選択した状態で、マウスの右クリックにてコンテクストメニューを 表示し、[スタートアッププロジェクトに設定(A)] を選択します。

ソリューション エクスプローラー		ビルド(U)
√∎ [™] © • ≒ 🗖 🗇 🗡 <mark>-</mark>		リビルド(E)
ソリューション エクスプローラー の検索		クリーン(N)
ק ソリューション 'TutorialSol' (4	Ξ	すべての子孫を折りたたむ
▲ 💷 BATCHRPT		ここまでスコープ指定する(S)
🎤 Properties	f	新しいソリューション Tクスプローラーのビュー(N)
Application1.mfgcf		
BATCHRPT.cbl		ビルドの依存関係(B)
EMPSEQ.cpy		追加(D)
ConsoleHello		
LoadCSVFile		
	.0	NuGet バッケージの管理(N)
ツリューション 1クスノローフー Git 変	£Ç;}	スタートアップ プロジェクトの構成
プロパティ		スタートアップ プロジェクトに設定(A)

メニューより、 [デバッグ(D)] > [ステップイン(I)] を選択するか F11 キーを押すと、コマンドプロンプト画面が開き、デバッガ ーがステップ実行を開始します。 デバッガーは手続き部の最初の COBOL 文である PERFORM 文を実行する手前で処 理を中断します。



デバ	ッグ(D)	テスト(S)	分析(N)	ツール(T)	拡張機能(X)	ウ
	ウィンド	ל(W)				•
	デバッグ	の開始(S)			F5	
\triangleright	デバッグ	なしで開始(ト	H)		Ctrl+F5	
2	パフォー	דעד דעד	イラー(F)		Alt+F2	
2	パフォー	ירםל געד	イラーの再起	動(L)		
°0	プロセス	にアタッチ(P)			Ctrl+Alt+P	
°0	プロセス	に再アタッチョ	13		Shift+Alt+P	
	その他の	Dデバッグ ター	ゲット(H)			•
	Micro I	Focus Code	Coverage T	で開始		
	Micro	Focus Profile	erで開始			
¥.	ステップ	イン(L)			F11	

BATCH	IRPT	EMPSEQ.cpy	BATCHRPT.cbl	+ ×	Program1.cbl	Fo
CBL BAT	CHRPT				👻 🖓 BAT	CHR
	- C	05 FILLER FIC X(1) 05 FILLER PIC X(5) 05 FILLER PIC X(7) 05 RPT-MSG PIC X(7) 05 FILLER PIC X(2) 05 FILLER PIC X(2) 05 RPT-TOT-RECS P 05 FILLER PIC X(3)) VALUE SPACE.) VALUE '***** 30) VALUE '***** 30) VALUE SPACE. 1C ZZZ. 3) VALUE SPACE	« '. Æ.		
₽	↓ C ▼ F	DI BLANK-LINE PIC X ROCEDURE DIVISION. PERFORM 1000-ST. PERFORM 2000-MA PERFORM 9000-CLI	(80) VALUE SPA ART THRU 1000- IN-PROCESSING DSE-AND-CLEANU	NCE. EXIT. THRU 2 JP THRU	2000-EXIT UNTIL AT J 9000-EXIT.	-EOF

制御ファイルから読み込んだレコードの内容を確認するため、データ部の CONTROL-REC 上でマウスの右クリックにてコン テクストメニューを表示し、[ウォッチの追加(W)]を選択します。

BATCHRPT	EMPSEQ.cpy		定義へ移動(G)	F12
BATCHRPT			すべての参照を検索(A)	Shift+F12
	88 NOT-AT-I	÷.	呼び出し階層の表示(H)	Ctrl+K, Ctrl+T
	10 EMP-REC		ブレークポイント(B)	•
	10 LINE-CH 10 LINE-MA 05 CURR-DATE 10 CURR-YY		データフロー グラフを表示 プログラムフロー グラフを表示	•
	10 CURR-MM 10 CURR-DD 05 CURR-TIME	\rightarrow	Code Analysis 	Alt+Num *
	10 CURR-HR 10 CURR-MI		カーソル行の前まで実行(N)	Ctrl+F10
Y	05 YRS-EMPLO 01 CONTROL-REC 05 CNTL-DATE	→	カーソルまで強制的に実行(O) 次のステートメントの設定(X)	Ctrl+Shift+F10
	10 CNTL-YR 10 CNTL-MOI 10 CNTL-DA		COBOL ウオッチポイントを追加	
►	** Employee Re 01 EMP-RECORD- COPY EMPSE	-	COBOL ブビックス フレークボイクトを追加 COBOL デバッグ ツールチップ スタイル	•
	** Report Line	60	ウォッチの追加(W)	

同様に入力ファイルから読み込んだレコードの内容を確認するため、データ部の EMP-RECORD-IO-AREA 上でマウス の右クリックにてコンテクストメニューを表示し、[ウォッチの追加(W)] を選択します。



BATCHRPT	EMPSEQ.cpy		定義へ移動(G)	F12
CBL BATCHRPT	Г]	すべての参照を検索(A)	Shift+F12
	88 NOT-AT-EOF VAL	\$	呼び出し階層の表示(H)	Ctrl+K, Ctrl+T
	10 EMP-REC-CNTR F		ブレークポイント(B)	•
	10 LINE-GIR PIC S		データフロー グラフを表示	•
	05 CURR-DATE.		プログラムフロー グラフを表示	
	10 CURR-MM PIC 90	Ś	Code Analysis	•
	05 CURR-TIME.	\rightarrow	次のステートメントの表示(H)	Alt+Num *
	10 CURR-HR PIC 90		カーソル行の前まで実行(N)	Ctrl+F10
	10 CURR-SEC PIC S		カーソルまで強制的に実行(O)	
Ļ	05 YRS-EMPLOYED PIC	1	次のステートメントの設定(X)	Ctrl+Shift+F10
i	05 CNTL-DATE.	C.	逆アセンブリへ移動(T)	Ctrl+K, G
	10 CNTL-MON PIC X	5	COBOL ウォッチポイントを追加	
-	10 CNIL-DAY PIC >	5	COBOL プログラム ブレークポイントを追加	
× ·	01 EMP-RECORD-IO-AREA		COBOL デバッグ ツールチップ スタイル	•
	** Report Lines	60	ウォッチの追加(W)	

前回のデバッグ作業で追加したウォッチ項目を削除します。画面下部より [ウォッチ 1] タブを選択し、[IN-REC] を選択、 マウスの右クリックでコンテクストメニューを開き、[ウォッチ式の削除(D)] を選択します。

ウォッチ 1		63	並列ウォッチの追加(R)
検索 (Ctrl+E)	 ← → 検索の詳細度: 	60) ウォッチ式の削除(<u>D</u>)
名前	值	-	すべて選択(<u>A</u>) Ctrl+A
😣 IN-REC	14-S 無効な作用対象がある	×	すべてクリア(<u>L</u>)
8 OUT-REC	14-S 無効な作用対象がある		トレーフポイントの挿入(T)
CONTROL-REC	{長さ=8}:""		
EMP-RECORD-IO-AREA 頂日をウォッズに迫加する	{長さ=60}: ************************************	0	16 進数で表示(<u>H</u>)
項目をワイッテに追加する		F	1 レベル上に折りたたむ(<u>O</u>)
自動 ローカル ウォッチ 1		1	ソース コードへ移動(G)

同様に、OUT-REC 項目もウォッチ式の削除を行ってください。

ウォッチ 1			→ ₽ ×
検索 (Ctrl+E)	 ← → 検索の詳細度: 		
名前	値	種類	
CONTROL-REC	{長さ=8}:" "	Q 表示 ▼ GROUP	
EMP-RECORD-IO-AREA	{長さ=60}: "	0 0 Q表示 ▼ GROUP	
項目をウォッチに追加する			
自動 ローカル ウォッチ 1			

手続き部 1000-START 節の READ 文に続く IF 文でエディター画面の左端をクリックし、ブレークポイントを設定します。





同様に手続き部 2000-MAIN-PROCESSING 段落の READ 文に続く IF 文でエディター画面の左端をクリックし、





メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押すと、デバッガーは最初のブレークポイントで実行を 中断します。

ウォッチ式の CONTROL-REC の値に制御ファイルから読み込んだレコードが表示されます。





再度、メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押すと、デバッガーは 2 番目のブレークポイント で実行を中断します。

ウォッチしている EMP-RECORD-IO-AREA の値に入力ファイルから読み込んだ1番目のレコードが表示されます。

BATCHRPT	EMPSEQ.cpy	BATCHRPT.cbl	⇔ × Pro	gram1.cbl		Form1.cbl		Form1.ct	ol [デザイ
BATCHRPT				•		IRPT			
v	WRITE RPT-RECOF WRITE RPT-RECOF 1個の参照 EXIT. 個の参照 2000-MAIN-PROCESSII READ EMP-SEO-F AT END MOVE '	NG. ILE INTO EMP-RE	INE BEFOR	Rea	ING 1 LI	INE. NE.			
100 % ▼ ● F	IF NOT-AT-EOF PERFORM 300 END-1F. (円の一共P2 問題は見つかりませんでした	00-PROCESS-RECC)RD THRU 30	DOO-EXIT					
ウォッチ 1								•	μ ×
検索 (Ctrl+E)	- م	$\leftarrow ightarrow$ 検索の	詳細度:	-					
名前		値					種類		
CONTROL-	REC	、 {長さ=8}: "20000	0101"			Q _{表示} ▼	GROUP		- 11
EMP-RECO	RD-IO-AREA	{長さ=60}: "1111	11113 佐藤	隆	<u> </u>	Q表示・	GROUP		
🔺 🤣 EMP-RE	С	{長さ=60}: "1111	11113 佐藤	隆	ዛ ኑሳ ዓክ	Q表示▼	GROUP		- 11
Semeral Employee	REC-SSN	11111113				Q表示▼	PIC X(8))	- 11
FILLE	R					Q表示▼	PIC X		- 11
Sector Employee	REC-JNAME1	佐藤				Q表示▼	PIC N(5)	-11
EMP	REC-JNAME2	隆					PIC N(5)	- 11
EMP	REC-NAME1	サトワ					PIC X(5))	-11
EMP		9/02 M)	- 11
	REC-GENDER	IVI							- 11
		告牲鸟)	- 1
	REC-DATE-OF-HIRE	白末中 (長さ=8)・"10080)401"			Q 表示 ▼	GROUP	/	-
自動 ローカル ウォ	ッチ1	(<u></u>), ,)))((-/3/11	5110 51		

BATCHRPT EMPSEQ.cpy	BATCHRPT.cbl 🗢 🗙 Prog	gram1.cbl	Form1.cbl	Form1.cbl [デザ
DEL BATCHRPT		✓ ^A gBA [*]	TCHRPT	
WRITE RPT-RECO	RD FROM RET-COLOMINS BEFORE RD FROM BLANK-LINE BEFORE	ADVANCING ADVANCING 1	LINE. LINE.	
1個の参照 1000-EXIT. EXIT. 1個の参照 2000-MAIN-PROCESSI READ EMP-SEQ-F AT END	NG. ILE INTO EMP-RECORD-IO-AR	EA		
MUVE IF NOT-AT-EOF PERFORM 301 END-IF. (END-#5R2	Y TU EUF-FLAG. 00-PROCESS-RECORD THRU 30	00-EXIT		
100 % - 🛛 🔊 問題は見つかりませんでした	t (
ウォッチ 1				- ₽ ×
検索 (Ctrl+E)	$\leftarrow ightarrow$ 検索の詳細度:	*		
名前	値		種	重類 ▲
CONTROL-REC	{長さ=8}: "20000101"		Q 表示 ▼ G	ROUP
EMP-RECORD-IO-AREA	{長さ=60}: "22222226 鈴木	尚之 スズキ	<mark>†</mark> Q 表示 ▼ G	ROUP
 MP-REC 	{長さ=60}: "22222226 鈴木	尚之 スズキ	<mark>†</mark> Q表示 ▼ G	ROUP
Semprec-SSN	2222226		Q表示 ▼ PI	IC X(8)
S FILLER			Q 表示 ▼ PI	IC X
Semprec-JNAME1	鈴木		Q 表示 ▼ PI	IC N(5)
Semprec-JNAME2	尚之		Q 表示 ▼ PI	IC N(5)
Semprec-Name1	77. \$		Q 表示 ▼ PI	IC X(5)
Semprec-NAME2	ŵ2		Q 表示 ▼ PI	IC X(5)
🤣 EMPREC-GENDER	M		Q 表示 ▼ PI	IC X
S FILLER			Q表示 ▼ PI	IC X
EMPREC-DIV	技術部		Q 表示 ▼ PI	IC N(5)
▶ 🤣 EMPREC-DATE-OF-HIRE 自動 ローカル <mark>ウォッチ 1</mark>	{長さ=8}: "19981015"		Q 表示 ▼ G	ROUP

再度、メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押すと、同じブレークポイント位置で停止しますが、2 番目のデータが表示されます。

ブレークポイントは、現在設定中の行の左端をクリックすることで解除できます。さきほど設定した2つのブレークポイントを解除してください。





メニューより、[デバッグ(D)] > [続行(C)] を選択するか F5 キーを押して、プログラムを終了します。

出力	
出力元(S): デバッグ - 🖕 🛀 🖆 😫 🔁	
スレッド、名前がありません、(12020)はコードの(0040)で終了しました。	
スレッド (名前かありません) (1516)はコード 0 (0x0)で終了しました。 スレッド (名前がありません) (4664)はコード 0 (0x0) で終了しました。	
スレッド、<名前がありません>*(1986)はコード0(0x0)で終了しました。 フレッド、<名前がありません>*(9459)はコード0(0x0)で終了しました。	- 4
スレッド '名前がありません>' (4024) はコード 0 (0x0) で終了しました。	
フロクラム [[]324] BAICHRMFLexe: c:#users#tarot#source#repos#tutorialsol#batchrpt#bin#xb4#debug#BAICHRMFLexe' ばコート U (UxU) で終了しました。	
• • • • • • • • • • • • • • • • • • •	- F
<u> ゴラー 第 出力</u>	

デバッグフォルダ (<3.2 節の 3) 「場所」で指定したフォルダ> ¥TutorialSol¥BATCHRPT¥bin¥x64¥debug) に Hire_Report.dat ファイルが作成されるので、メモ帳などのエディターでファイルを開き、社員 3 名分のデータが表示され ることを確認します。



📃 Cnt	I_Card.dat			Hire_Report.dat	×	+	
ファイル	編集	表示					
Program	: BATCHR	PT	Years En	nployed Report		08/26/2	024
***>	** 200C	年 1月 1日じ	↓前に入社!	した社員一覧		10.00	
部署	8名	社員名		社員番号	入社日	雇用	年数
営業 技術 総務	警察 19 19 19 19 19 19 19 19 19 19 19 19 19	佐藤 鈴木 田中	隆 尚之 直美	1111111-3 2222222-6 33333333-9	04/01/1998 10/15/1998 04/01/1999	26 26 25	
***>	** 処理	レコード件数	汝:	3			

続いて、基準日により結果が変更されることを確認します。

デバッグフォルダ (<3.2 節の 3) 「場所」で指定したフォルダ> ¥TutorialSol¥BATCHRPT¥bin¥x64¥debug) に Cntl_Card.dat ファイルの中身を 20110101 に更新します。

	Cntl_Card.da	t	×	+
ファイル	編集	表示		
2011010	01			

メニューより、[デバッグ(D)] > [デバッグの開始(S)] を選択するか F5 キーを押してプログラムを実行します。 今は、ブレー クポイントが設定されていないため、そのままプログラムは正常終了します。

デバ	、ッグ(D)	テスト(S)	分析(N)	ツール(T)	拡張機能(X)	ļ,	
ウィンドウ(W)							
	デバッグ	の開始(S)			F5		
\triangleright	デバッグ	なしで開始(I	H)		Ctrl+F5		

出力元(S): デバッグ - ― 二 二 二 二 二 四
スレッド '<名前がありません>'(12020)はコード 0 (0x0)で終了しました。 スレッド '<名前がありません>'(8092)はコード 0 (0x0)で終了しました。 スレッド '<名前がありません>'(12768)はコード 0 (0x0)で終了しました。 スレッド '<名前がありません>'(12604)はコード 0 (0x0)で終了しました。
ブログラム '[12428] BATCHRPT.exe: c:¥users¥tarot¥source¥repos¥tutorialsol¥batchrpt¥bin¥x64¥debug¥BATCHRPT.exe'はコード 0 (0x0) で終了しました。



さきほど確認した Hire_Report.dat ファイルが更新され、2011 年 1 月 1 日以前に入社した社員 9 名分のデータが 表示されることを確認します。

Cntl_Card.c	lat		Hire_Report.dat		× +
ファイル 編集	表示				
Program: BATC ***** 20	CHRPT)11年 1月 1E	Years Ei 日以前に入社	mployed Report した社員一覧		08/26/2024 16:08:44
部署名	社員名		社員番号	入社日	雇用年数
部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部部	佐鈴田山伊木中橋三藤木中田藤村村本井	隆尚直洋弘貴慎悦薫	1111111-3 2222222-6 33333333-9 444444-2 5555555-5 6666666-8 7777777-1 8888888-4 9999999-7	04/01/1998 10/15/1998 04/01/1999 07/01/2000 04/01/2001 12/20/2002 04/01/2003 08/05/2004 04/01/2005	26 25 24 22 22 21 20 19
*****	L理レコード化	牛数:	9		

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。 ソースコード、説明、その他すべてについて、無謬性は保障されません。 ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。 本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。